

## Oracle 11g RMAN: BZIP2 vs. ZLIB

Prior to 11g Oracle RMAN had a single compression algorithm, called BZIP2. The algorithm has a very satisfactory compression ratio in terms of decreasing the size of RMAN output. However, high CPU cost makes algorithm not suitable for many sites especially for sites having CPU bottleneck (Data warehouse DBAs?!?☺). As a result people still use hardware compression capabilities of tape drivers (ratios like 1:3) to decrease the backup time and increase the effective write speed of backup drivers.

By 11g Oracle introduces a new compression algorithm that is announced to be less compressive but less aggressive in terms of CPU. In this paper you will find comparison of two algorithms with no compressed case.

## How to Use Compression in RMAN

Remember that not all RMAN backups can be compressed. This is one of the most famous fallacies about RMAN. For example you cannot compress image backups. But you can compress data file BACKUPSET and archive log BACKUPSET backups. This paper only covers compression of data file BACKUPSET backups in detail.

## Compress or Not to Compress

Creating a compressed backup is simply achieved by **COMPRESSED BACKUP** clause in RMAN. In order to create compressed archive log backupsets, use

```
RMAN> backup as compressed backupset archivelog all;
```

```
Starting backup at 03-SEP-08 13:10:28
current log archived
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=64 device type=DISK
allocated channel: ORA_DISK_2
channel ORA_DISK_2: SID=75 device type=DISK
allocated channel: ORA_DISK_3
channel ORA_DISK_3: SID=37 device type=DISK
allocated channel: ORA_DISK_4
channel ORA_DISK_4: SID=74 device type=DISK
channel ORA_DISK_1: starting compressed archived log backup set
channel ORA_DISK_1: specifying archived log(s) in backup set
input archived log thread=1 sequence=3590 RECID=3530 STAMP=664463438
channel ORA_DISK_1: starting piece 1 at 03-SEP-08 13:11:10
channel ORA_DISK_2: starting compressed archived log backup set
channel ORA_DISK_2: specifying archived log(s) in backup set
input archived log thread=1 sequence=3589 RECID=3529 STAMP=664438725
channel ORA_DISK_2: starting piece 1 at 03-SEP-08 13:11:25
channel ORA_DISK_1: finished piece 1 at 03-SEP-08 13:11:25
piece handle=+RGRP/ods/backupset/2008_09_03/annnf0_tag20080903t131055_0.1188.664463471
tag=TAG20080903T131055 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:15
channel ORA_DISK_2: finished piece 1 at 03-SEP-08 13:11:28
piece handle=+RGRP/ods/backupset/2008_09_03/annnf0_tag20080903t131055_0.1189.664463485
tag=TAG20080903T131055 comment=NONE
channel ORA_DISK_2: backup set complete, elapsed time: 00:00:03
Finished backup at 03-SEP-08 13:11:28
```

In order to create compressed data file backupsets of a number of tablespaces' data files use

```
RMAN> backup as compressed backupset tablespace owb_repos, tools, users, system, sysaux, undotbs1;
```

```
Starting backup at 02-SEP-08 16:07:29
using channel ORA_DISK_1
using channel ORA_DISK_2
channel ORA_DISK_1: starting compressed full datafile backup set
```

```

channel ORA_DISK_1: specifying datafile(s) in backup set
input datafile file number=00003 name=+DGRP/ods/datafile/undotbs1.262.661444779
input datafile file number=00004 name=+DGRP/ods/datafile/users.264.661444793
input datafile file number=00007 name=+DGRP/ods/datafile/tools.283.661616945
channel ORA_DISK_1: starting piece 1 at 02-SEP-08 16:07:45
channel ORA_DISK_2: starting compressed full datafile backup set
channel ORA_DISK_2: specifying datafile(s) in backup set
input datafile file number=00002 name=+DGRP/ods/datafile/sysaux.261.661444777
input datafile file number=00006 name=+DGRP/ods/datafile/owb_repos.263.661556151
input datafile file number=00001 name=+DGRP/ods/datafile/system.260.661444773
channel ORA_DISK_2: starting piece 1 at 02-SEP-08 16:08:00
channel ORA_DISK_2: finished piece 1 at 02-SEP-08 16:10:46
piece handle=+RGRP/ods/backupset/2008_09_02/nnndf0_tag20080902t160730_0.295.664387681
tag=TAG20080902T160730 comment=NONE
channel ORA_DISK_2: backup set complete, elapsed time: 00:02:46
channel ORA_DISK_1: finished piece 1 at 02-SEP-08 16:26:57
piece handle=+RGRP/ods/backupset/2008_09_02/nnndf0_tag20080902t160730_0.298.664387665
tag=TAG20080902T160730 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:19:12
Finished backup at 02-SEP-08 16:26:57

```

## How to Change Compression Algorithms

Once you check in RMAN console, you will see that default RMAN compression algorithm is good old BZIP2

```
RMAN> show all;
```

```

RMAN configuration parameters for database with db_unique_name ODS are:
CONFIGURE RETENTION POLICY TO REDUNDANCY 1; # default
CONFIGURE BACKUP OPTIMIZATION ON;
CONFIGURE DEFAULT DEVICE TYPE TO DISK; # default
CONFIGURE CONTROLFILE AUTOBACKUP ON;
CONFIGURE CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE DISK TO '%F'; # default
CONFIGURE DEVICE TYPE DISK PARALLELISM 2 BACKUP TYPE TO BACKUPSET;
CONFIGURE DATAFILE BACKUP COPIES FOR DEVICE TYPE DISK TO 1; # default
CONFIGURE ARCHIVELOG BACKUP COPIES FOR DEVICE TYPE DISK TO 1; # default
CONFIGURE MAXSETSIZE TO UNLIMITED; # default
CONFIGURE ENCRYPTION FOR DATABASE OFF; # default
CONFIGURE ENCRYPTION ALGORITHM 'AES128'; # default
CONFIGURE COMPRESSION ALGORITHM 'BZIP2'; # default
CONFIGURE ARCHIVELOG DELETION POLICY TO NONE; # default
CONFIGURE SNAPSHOT CONTROLFILE NAME TO '/u01/app/oracle/product/11.1.0/ods/dbs/snapcf_ods.f'; # default

```

You can simply change the compression algorithm to ZLIB.

```

RMAN> CONFIGURE COMPRESSION ALGORITHM 'ZLIB';

new RMAN configuration parameters:
CONFIGURE COMPRESSION ALGORITHM 'ZLIB';
new RMAN configuration parameters are successfully stored

```

Or you can replace algorithm back to BZIP2.

```

RMAN> CONFIGURE COMPRESSION ALGORITHM 'BZIP2';

old RMAN configuration parameters:
CONFIGURE COMPRESSION ALGORITHM 'ZLIB';
new RMAN configuration parameters:
CONFIGURE COMPRESSION ALGORITHM 'BZIP2';
new RMAN configuration parameters are successfully stored

```

## Active Session History (ASH) Results

I have compared the RMAN session's wait event and CPU times by performing a two channel RMAN backup to flash recovery area on an ASM database. Keep in mind that the system I have performed the tests is completely a disk limiting system. It has computation power abundance.

As you will see below, ASH results **BZIP2** algorithm consumes **x25 more CPU** time whereas **ZLIB** consumes just **x15 more CPU** with compared to uncompressed backup.

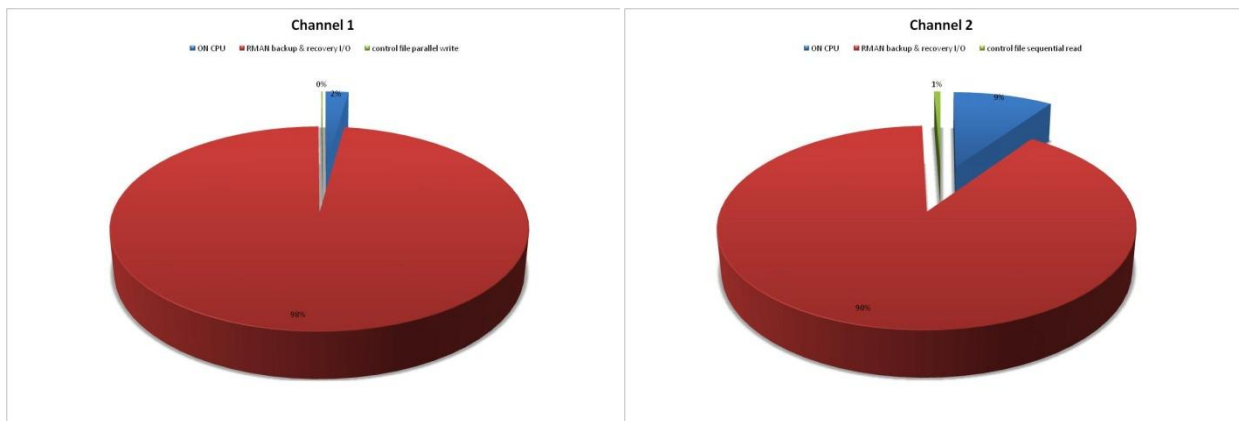
In terms of RMAN I/O waits, **BZIP2** and **ZLIB** seem to be very close to each other. The gap is statistically insignificant. Their **I/O wait is 80%** of uncompressed backup scenario.

In total there is a gap between the backup duration of **BZIP2** compressed backup and **ZLIB** compressed backups. While **BZIP2** takes more than **25 minutes**, **ZLIB** takes only **19 minutes**. With compared to uncompressed scenario (**14 minutes**), **BZIP2** seems to be unacceptable in terms of backup time where as **ZLIB** is tolerable.

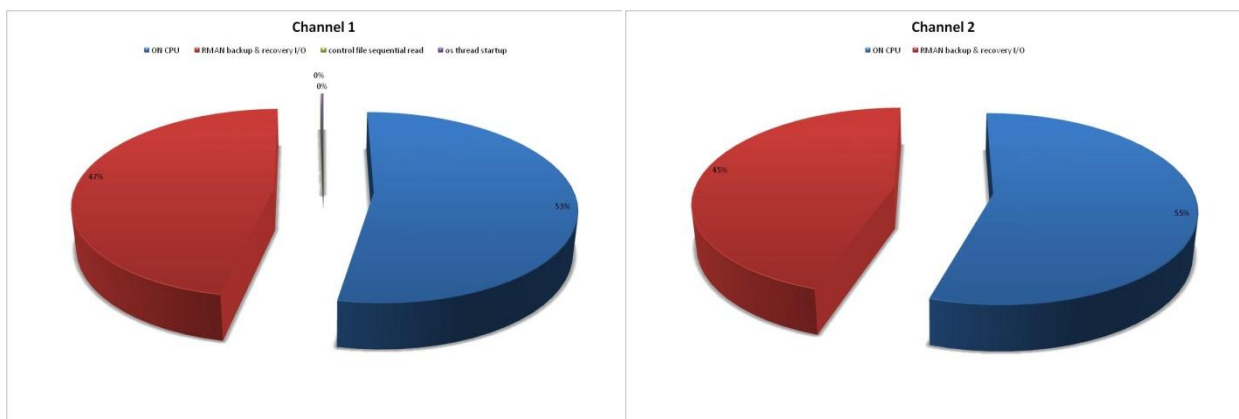
	NoCompression				BZIP2				ZLIB			
	ON CPU	RMAN backup & recovery I/O	Others	Total	ON CPU	RMAN backup & recovery I/O	Others	Total	ON CPU	RMAN backup & recovery I/O	Others	Total
Channel 1	19	850	1	870	803	706	4	1513	480	671	0	1151
Channel 2	16	151	1	168	122	100	0	222	55	120	0	175
Total	35	1001	2	1038	925	806	4	1735	535	791	0	1326

Good  
Normal  
Bad

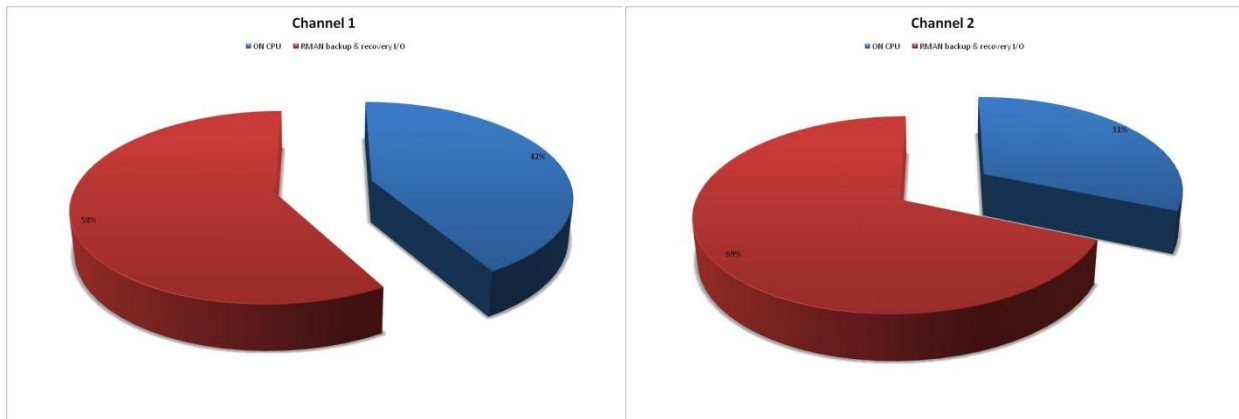
Result 1 ASH Output for RMAN Sessions



Result 2 ASH Wait + CPU Distribution for NOCOMPRESS Backup



Result 3 ASH Wait + CPU Distribution for BZIP2 Backup



Result 4 ASH Wait + CPU Distribution for ZLIB Backup

### V\$BACKUP\_\*\_IO Results

Two views **V\$BACKUP\_ASYNC\_IO** and **V\$BACKUP\_SYNC\_IO** are my favorite views when it comes to **RMAN**. Let's look at their results

SID	T	FILENAME	TOTAL_MB	DURATION	MB	MB/S	IO_COUNT	READY
73	A			86900	13006,65	14,96	14921	10843 (72,6%)
73	I	+DGRP/ods/datafile/undotbs1.262.661444779	30330	86900	12821	14,75	14729	10724 (72,8%)
73	I	+DGRP/ods/datafile/users.264.661444793	175,56	2000	167,07	8,35	170	105 (61,7%)
73	I	+DGRP/ods/datafile/tools.283.661616945	100	400	18,57	4,64	22	14 (63,6%)
73	O	+RGRP		87000	13004,87	14,94	13005	4649 (35,7%)
74	A			15300	2497,23	16,32	2507	1609 (64,1%)
74	I	+DGRP/ods/datafile/sysaux.261.661444777	1801,18	15300	1715,57	11,21	1719	1141 (66,3%)
74	I	+DGRP/ods/datafile/owb_repos.263.661556151	1024	100	0,07	0,07	3	1 (33,3%)
74	I	+DGRP/ods/datafile/system.260.661444773	790	8900	781,57	8,78	785	467 (59,4%)
74	O	+RGRP		15300	2151,43	14,06	2152	354 (16,4%)

Result 5 UNCOMPRESSED Backup Results

Notice that the aggregate read size is almost equal to output write size for UNCOMPRESSED backup. Moreover the percentage of **READY** during backup write is low. This ratio is an important indicator of disk bottleneck. RMAN can't fill an output buffer, because IO subsystem stops it to fill.

SID	T	FILENAME	TOTAL_MB	DURATION	MB	MB/S	IO_COUNT	READY
73	A			151000	13004,65	8,61	14921	11863 (79,5%)
73	I	+DGRP/ods/datafile/undotbs1.262.661444779	30330	151000	12819	8,48	14729	11716 (79,5%)
73	I	+DGRP/ods/datafile/users.264.661444793	175,56	3000	167,07	5,56	170	133 (78,2%)
73	I	+DGRP/ods/datafile/tools.283.661616945	100	400	18,57	4,64	22	14 (63,6%)
73	O	+RGRP		151000	3477,64	2,3	3478	2106 (60,5%)
74	A			20800	2497,23	12	2507	1993 (79,4%)
74	I	+DGRP/ods/datafile/sysaux.261.661444777	1801,18	20800	1715,57	8,24	1719	1368 (79,5%)
74	I	+DGRP/ods/datafile/owb_repos.263.661556151	1024	0	0,07	0,07	3	1 (33,3%)
74	I	+DGRP/ods/datafile/system.260.661444773	790	12100	781,57	6,45	785	624 (79,4%)
74	O	+RGRP		20900	300,5	1,43	301	190 (63,1%)

Result 6 BZIP2 Backup Results

**BZIP2** backup significantly compressed the input and reduced the size of backup output by 1:4 – 1:8. Moreover notice that the **READY** percentage for write operations significantly increased. This is obviously because RMAN has fewer things to write to disk.

SID	T	FILENAME	TOTAL_MB	DURATION	MB	MB/S	IO_COUNT	READY
73	A			114700	13005	11,33	14921	11848 (79,4%)
73	I	+DGRP/ods/datafile/undotbs1.262.661444779	30330	114700	12819	11,17	14729	11700 (79,4%)
73	I	+DGRP/ods/datafile/users.264.661444793	175,56	2100	167,07	7,95	170	134 (78,8%)
73	I	+DGRP/ods/datafile/tools.283.661616945	100	400	18,57	4,64	22	14 (63,6%)
73	O	+RGRP		114800	4322,2	3,76	4323	2491 (57,6%)
74	A			16100	2498,2	15,51	2508	1994 (79,5%)
74	I	+DGRP/ods/datafile/sysaux.261.661444777	1801,18	16100	1716,6	10,66	1720	1369 (79,5%)
74	I	+DGRP/ods/datafile/owb_repos.263.661556151	1024	100	0,07	0,07	3	1 (33,3%)
74	I	+DGRP/ods/datafile/system.260.661444773	790	9400	781,57	8,31	785	624 (79,4%)
74	O	+RGRP		16200	341,62	2,1	342	210 (61,4%)

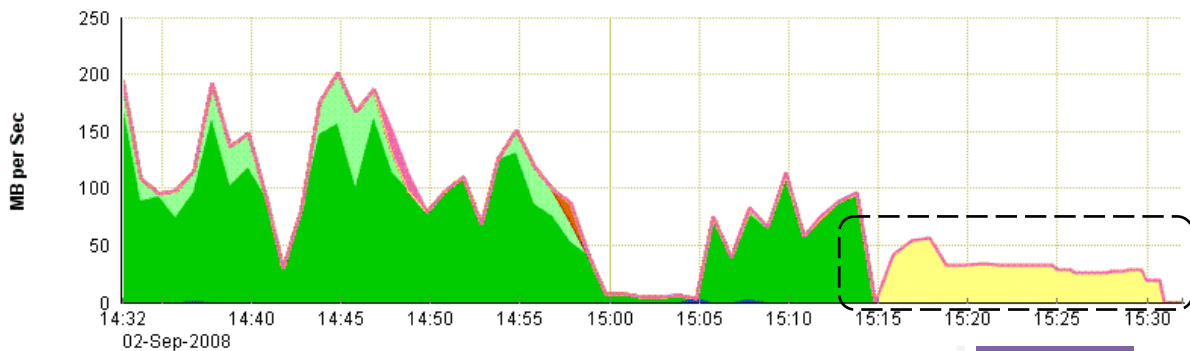
Result 7 ZLIB Backup Results

ZLIB also seems to compress the data significantly. As you may expect compression ratio is less than BZIP2 compression ratio. It is around 1:3 – 1:7. Again READY percentage is significantly higher than UNCOMPRESSED backup due to same reason.

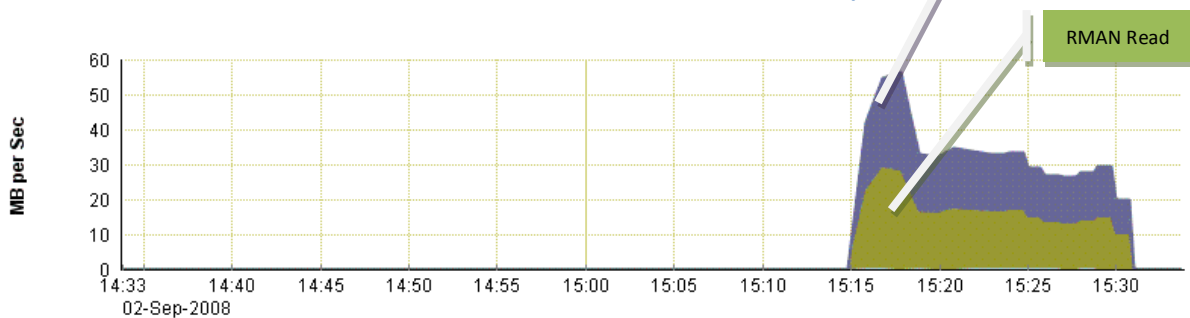
This shows that the percentage of IO waits in ASH report is highly due to decreased write IO.

Seeing Reduced Write IO by EM Graphs

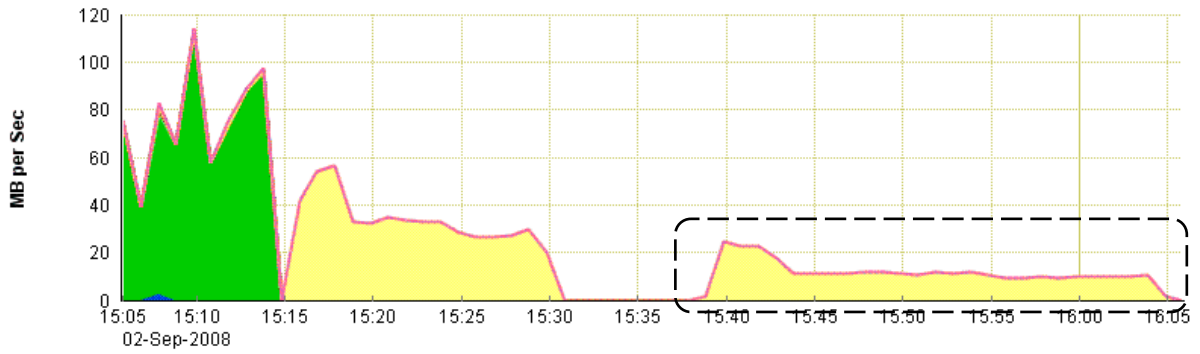
By 11g Oracle starts to measure IO metrics in a detailed fashion also it illustrates those metrics via EM graphs. Let's finally see the reduced disk I/O from enterprise manager windows.



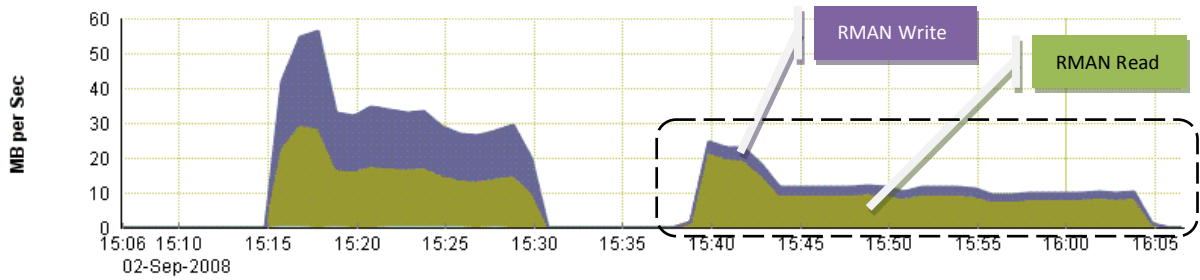
Result 8 Total RMAN IO for UNCOMPRESSED Backup



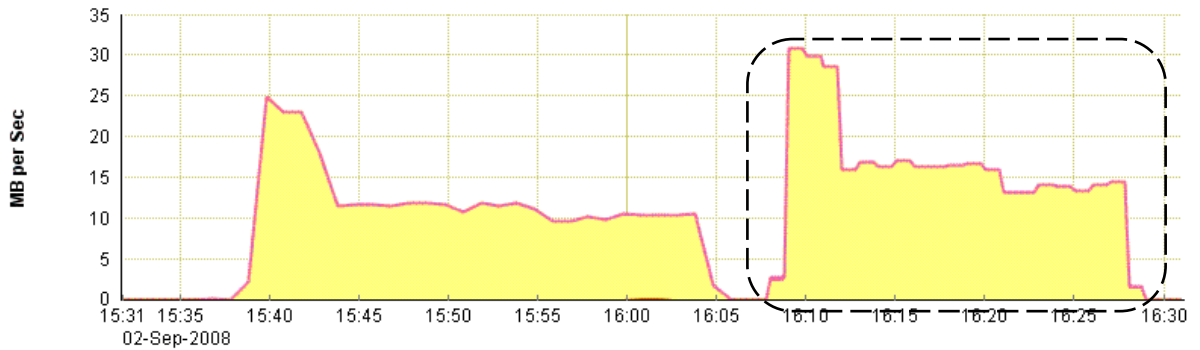
Result 9 Factoring RMAN IO for UNCOMPRESSED Backup



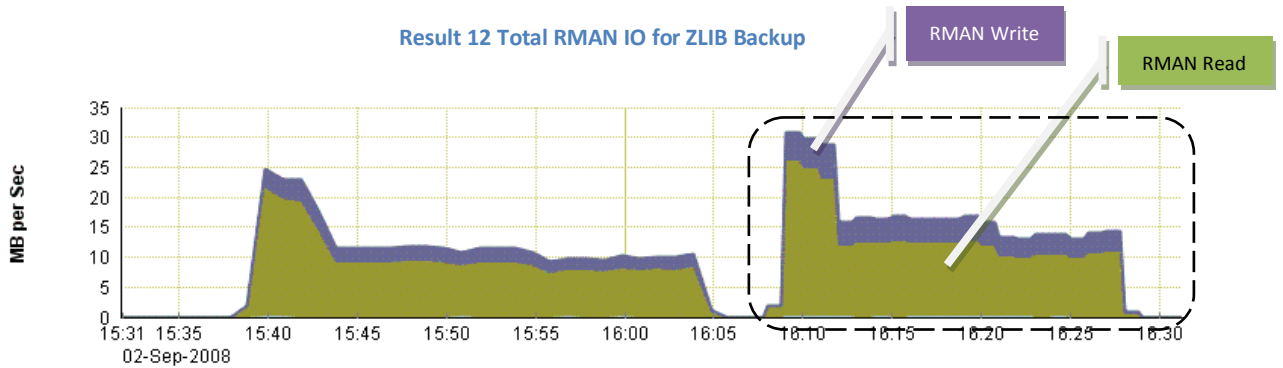
Result 10 Total RMAN IO for BZIP2 Backup



Result 11 Factoring RMAN IO for BZIP2 Backup



Result 12 Total RMAN IO for ZLIB Backup



Result 13 Factoring RMAN IO for ZLIB Backup

## Conclusion

Brand new **ZLIB** algorithm really gives what it is told to give. Significantly reduced CPU consumption comes with tolerable compression ratio reduction. Keep in mind that although it is reduced with compared to **BZIP2** algorithm, the time elapsed on CPU for **ZLIB** algorithm is still far beyond the time required for UNCOMPRESSED scenario.

After all I can conclude about the usage of this algorithm when the following conditions are hold:

1. If your system is not in CPU bottleneck in backup periods.
2. If you are taking incremental backup or your database size is a few hundred gigabytes. Once the size of your database increases, the gap between UNCOMPRESSED backup and ZLIB compressed backup performance will grow. But if you use incremental backup strategy (remember that incrementally updated backup is still the advised technique), the size of backupsets will be small with compared to total size of the database which make the use of ZLIB acceptable.
3. If your backups are on a low or middle cost disk tier. Remember that compression is beneficial because it reduces the amount of data that is to be written. The lower the IO on low cost tier, the lower your pain in backup durations.
4. If you use tape based backup ensure that tape compression is disable. The tape compression with RMAN compression is not an advised way of doing job.

## Scripts

I have attached the scripts I have used to monitor the performance of RMAN backups.

### Active Session History (ASH) Results

```
select session_id, nvl(event, session_state) event, count(*) sec
  from v$active_session_history
 where session_id in (&rman_session_1_id, &rman_session_2_id)
    and sample_time between
        to_date('&backup_start', '&suitable_mask') and
        to_date('&backup_end', '&suitable_mask')
 group by session_id, nvl(event, session_state)
 order by session_id;
```

### V\$BACKUP\_\*\_IO Results

```
select  sid,
        substr(type,1,1) T,
        filename,
        buffer_count bc,
        trunc(total_bytes / 1024 / 1024,2) total_mb,
        elapsed_time duration,
        maxopenfiles mof,
        trunc(bytes / 1024 / 1024,2) mb,
        trunc(effective_bytes_per_second / 1024 / 1024,2) "MB/S",
        io_count,
        ready || ' (' || trunc(ready * 100 / io_count, 1) || '%)' ready,
        short_waits || ' (' || trunc(short_waits * 100 / io_count, 1) ||
        '%) - ' || SHORT_WAIT_TIME_TOTAL || ' sn' short_wait,
        long_waits || ' (' || trunc(long_waits * 100 / io_count, 1) ||
        '%) - ' || LONG_WAIT_TIME_TOTAL || ' sn' long_wait
  from v$backup_async_io
 where open_time between
        to_date('&backup_start', '&suitable_mask') and
        to_date('&backup_end', '&suitable_mask')
    and close_time between
        to_date('&backup_start', '&suitable_mask') and
        to_date('&backup_end', '&suitable_mask')
 order by sid, type;
```